UNITED STATES DEPARTMENT OF COMMERCE
National Bureau of Standards
Washington, D.C. 20234

ODP # _171276_

FIPS

1976 August 30

MEMORANDUM FOR FIPS Points of Contact
                              FIPSCAC

From: Harry S. White, Jr.   Harry S. White Jr.
        Associate Director for ADP Standards

Subject: Review and Comment on BSR - X3.9 Draft Proposed ANS FORTRAN

My memorandum of 1976 July 30 provided for your review and comment
the draft proposed American National Standard FORTRAN (BSR - X3.9).

I have just received the attached press release from ANSI concerning
the review process on the draft proposal and recent changes made by
the responsible technical standards committee (X3J3).

This additional material is for your information and use in preparing
appropriate comments to the X3J3 Committee Secretary, Mr. Lloyd Campbell,
BRL-CSD, Building 328, Aberdeen Proving Ground, Maryland 21005.

Attachment

C15 CM A

COMMITTEE CORRESPONDENCE

PRESS RELEASE

"FORTRAN Standards Committee Adopts IF-THEN-ELSE"

The FORTRAN Standards Committee met in Idaho Falls, Idaho during July 12-15 to begin reviewing public comments received on the draft proposed revised FORTRAN standard. The committee, also known as X3J3, is a technical committee of the American National Standards Institute (ANSI).

At the meeting, X3J3 approved the addition of four new statements that together provide the capability to conditionally execute groups of statements. They are called block IF, ELSE IF, ELSE and END IF statements. The need for this capability was strongly presented in many of the public comments. It was also a lively topic of discussion at two public presentations on the draft standard that took place in Los Angeles in February and Washington, D.C. in March.

X3J3 published its draft proposal in the March issue of SIGPLAN Notices, a publication of the Special Interest Group on Programming Languages of the Association for Computing Machinery. More than eight thousand copies have been distributed to interested individuals, and technical, business and governmental organizations around the world.

The widespread interest in the proposal for a revised FORTRAN standard is indicated by the substantial volume of comments received. As of the beginning of the meeting, 200 letters had been received totaling 810 pages. The over- whelming majority of comments are favorable and contain many constructive suggestions. According to ANSI procedures, each suggestion will be evaluated to determine whether a change should be made to the draft standard. Following completion of the X3J3 review process, each public review letter will be answered indicating the action taken.

X3J3 will continue its review of public comments at its next meeting in September. The public review and comment period closes September 28, 1976.

(Approved for release by
X3J3 on 15 July 1976)

Bell Laboratories

11. CONTROL STATEMENTS   1

To X3J3:

IF-THEN-ELSE was adopted at the July meeting of X3J3. The attached press release and IF-THEN-ELSE text is being sent to you as information relating to the processing of dpANS FORTRAN.

Press Release

Attached is a press release announcing the adoption of IF-THEN-ELSE by X3J3. IF-THEN-ELSE was adopted for the FORTRAN full language and the subset language.

IF-THEN-ELSE Text

The principal change to the dpANS FORTRAN is to Section 11, CONTROL STATEMENTS. Section 11 of Document X3J3/76.3 FORTRAN Full Language is attached. The text of the subset is not attached since the IF-THEN-ELSE subset text is identical to that of the full language.

Document X3J3/76 remains the basis document for dpANS FORTRAN. Document X3J3/76.3 is a working document of X3J3 and is subject to further changes.

Comments on dpANS FORTRAN or the new IF-THEN-ELSE text may be sent to:

Lloyd W. Campbell
X3J3 Secretary
BRL-CSD Bldg. 328
Aberdeen Proving Ground
MD 21005 USA

HO-8223-JCN-dg

J. C. Noll

Att.
Press Release
X3J3/76.3 Section 11,
 CONTROL STATEMENTS

RECEIVED

AUG 2 1976

Control statements may be used to control the execution   3
sequence.   4

There are sixteen control statements: 6

   (1) Unconditional GO TO   8

   (2) Computed GO TO   10

   (3) Assigned GO TO   12

   (4) Arithmetic IF   14

   (5) Logical IF   16

   (6) Block IF   18

   (7) ELSE IF   20

   (8) ELSE   22

   (9) END IF   24

   (10) DO   26

   (11) CONTINUE   28

   (12) STOP   30

   (13) PAUSE   32

   (14) END   34

   (15) CALL   36

   (16) RETURN   38

CALL and RETURN statements are described in Section 15.   40

11.1 Unconditional GO TO Statement   43

The form of an unconditional GO TO statement is: 45

      GO TO s 47

where s is the statement label of an executable statement   49
that appears in the same program unit as the unconditional   50
GO TO statement.   51

Execution of an unconditional GO TO statement causes a   53
transfer of control so that the statement identified by the   54
statement label is executed next.   55

Page 11-2 CONTROL STATEMENTS

11.2 Computed GO TO Statement   61

The form of a computed GO TO statement is: 63

        GO TO (s [,s]...) [,] e   65

where: e is an integer, real, or double precision   67
          expression   68

       s is the statement label of an executable statement   70
          that appears in the same program unit as the   71
          computed GO TO statement. The same statement   72
          label may appear more than once in the same   73
          computed GO TO statement.   74

Execution of a computed GO TO statement causes evaluation of   76
the expression INT(e). Let the value of INT(e) be i. The   77
evaluation of INT(e) is followed by a transfer of control so   78
that the statement identified by the ith statement label in   79
the list of statement labels is executed next, provided that   80
1 ≤ i ≤ n, where n is the number of statement labels in the   81
list of statement labels. If i<1 or i>n, the execution   82
sequence continues as though a CONTINUE statement were   83
executed.   84

11.3 Assigned GO TO Statement   87

The form of an assigned GO TO statement is: 89

        GO TO i [[,] (s [,s]...)]   91

where: i is an integer variable name   93

       s is the statement label of an executable statement   95
          that appears in the same program unit as the   96
          assigned GO TO statement. The same statement   97
          label may appear more than once in the same   98
          assigned GO TO statement.   99

At the time of execution of an assigned GO TO statement, the   101
current value of i must have been assigned by the prior   102
execution of an ASSIGN statement (10.3) to the statement   103
label of an executable statement. The execution of the   104
assigned GO TO statement causes a transfer of control so   105
that the statement identified by that statement label is   106
executed next. The last definition of the variable in an   107
assigned GO TO statement must have occurred in the same   108
program unit as the assigned GO TO statement.   109

If the parenthesized list is present, the statement label   111
assigned to i must be one of the statement labels in the   112
list.   113

FORTRAN/76 Full Language X3J3/76.3 (76-07-19)

CONTROL STATEMENTS Page 11-3

11.4 Arithmetic IF Statement   121

The form of an arithmetic IF statement is: 123

        IF (e) s₁ , s₂ , s₃   125

where: e is an integer, real, or double precision   127
          expression   128

       s₁, s₂, and s₃ are each the statement label of an   130
          executable statement that appears in the same   131
          program unit as the arithmetic IF statement. The   132
          same statement label may appear more than once in   133
          the same arithmetic IF statement.   134

Execution of an arithmetic IF statement causes evaluation of   136
the expression e followed by a transfer of control. The   137
statement identified by s₁, s₂, or s₃ is executed next as   138
the value of e is less than zero, equal to zero, or greater   139
than zero, respectively.   140

11.5 Logical IF Statement   143

The form of a logical IF statement is: 145

        IF (e) st   147

where: e is a logical expression   149

       st is any executable statement except a DO,   151
          block IF, ELSE IF, ELSE, END IF, END, or another   152 |
          logical IF statement.   153 |

Execution of a logical IF statement causes evaluation of the   155
expression e. If the value of e is true, statement st is   156
executed. If the value of e is false, statement st is not   157
executed and the execution sequence continues as though a   158
CONTINUE statement were executed.   159

Note that the execution of a function reference in the   161
expression e of a logical IF statement is permitted to   162
affect entities in the statement st.   163

11.6 Block IF Statement   166 |

The block IF statement is used with the END IF statement   168 |
and, optionally, the ELSE IF and ELSE statements to control   169 |
the execution sequence.   170 |

The form of a block IF statement is: 172 |

        IF (e) THEN   174 |

where e is a logical expression.   176 |

X3J3/76.3 (76-07-19) FORTRAN/76 Full Language

11.6.1 IF-level   181 |

The IF-level of a statement s is   183 |

$$n_1 - n_2$$ 185 |

where n_1 is the number of block IF statements from the   187 |
beginning of the program unit up to and including s, and n_2 188 |
is the number of END IF statements in the program unit up to   189 |
but not including s.   190 |

The IF-level of every statement must be zero or positive.   192 |
The IF-level of each block IF, ELSE IF, ELSE, and END IF   193 |
statement must be positive. The IF-level of the END   194 |
statement of each program unit must be zero.   195 |

11.6.2 IF-Block   197 |

An IF-block consists of all of the executable statements   199 |
after the block IF statement up to, but not including, the   200 |
next ELSE IF, ELSE, or END IF statement that has the same   201 |
IF-level as the block IF statement. An IF-block may be   202 |
empty.   203 |

11.6.3 Execution of a Block IF Statement   205 |

Execution of a block IF statement causes evaluation of the   207 |
expression e. If the value of e is true, normal execution   208 |
sequence continues with the first statement of the IF-block.   209 |
If the IF-block is empty, control is transferred to the next   210 |
END IF statement that has the same IF-level as the block IF   211 |
statement. If the value of e is false, control is   212 |
transferred to the next ELSE IF, ELSE, or END IF statement   213 |
that has the same IF-level as the block IF statement.   214 |

Transfer into an IF-block is permitted.   216 |

If the execution of the last statement in the IF-block does   218 |
not result in a transfer of control, control is transferred   219 |
to the next END IF statement that has the same IF-level as   220 |
the block IF statement that precedes the IF-block.   221 |

11.7 ELSE IF Statement   224 |

The form of an ELSE IF statement is: 226 |

        ELSE IF (e) THEN   228 |

where e is a logical expression.   230 |

11.7.1 ELSE IF-Block   232 |

An ELSE IF-block consists of all of the executable   234 |
statements after the ELSE IF statement up to, but not   235 |
including, the next ELSE IF, ELSE, or END IF statement that   236 |

has the same IF-level as the ELSE IF statement. An ELSE IF-   241 |
block may be empty.   242 |

11.7.2 Execution of an ELSE IF Statement   244 |

Execution of an ELSE IF statement causes evaluation of the   246 |
expression e. If the value of e is true, normal execution   247 |
sequence continues with the first statement of the ELSE IF-   248 |
block. If the ELSE IF-block is empty, control is   249 |
transferred to the next END IF statement that has the same   250 |
IF-level as the ELSE IF statement. If the value of e is   251 |
false, control is transferred to the next ELSE IF, ELSE, or   252 |
END IF statement that has the same IF-level as the ELSE IF   253 |
statement.   254 |

Transfer into an ELSE IF-block is permitted.   256 |

If execution of the last statement in the ELSE IF-block does   258 |
not result in a transfer of control, control is transferred   259 |
to the next END IF statement that has the same IF-level as   260 |
the ELSE IF statement that precedes the ELSE IF-block.   261 |

11.8 ELSE Statement   264 |

The form of an ELSE statement is: 266 |

        ELSE   268 |

11.8.1 ELSE-Block   270 |

An ELSE-block consists of all of the executable statements   272 |
after the ELSE statement up to, but not including, the next   273 |
END IF statement that has the same IF-level as the ELSE   274 |
statement. An ELSE-block may be empty.   275 |

An END IF statement of the same IF-level as the ELSE   277 |
statement must appear before the appearance of an ELSE IF or   278 |
ELSE statement of the same IF-level.   279 |

11.8.2 Execution of an ELSE Statement   281 |

Execution of an ELSE statement has no effect. Normal   283 |
execution sequence continues.   284 |

Transfer into an ELSE-block is permitted.   286 |

11.9 END IF Statement   289 |

The form of an END IF statement is: 291 |

        END IF   293 |

Execution of an END IF statement has no effect. Normal   295 |
execution sequence continues.   296 |

Page 11-6 CONTROL STATEMENTS

For each block IF statement, there must be a corresponding 301
END IF statement in the same program unit. A corresponding 302
END IF statement is the next END IF statement that has the 303
same IF-level as the block IF statement. 304

11.10 DO Statement 307

A DO statement is used to specify a loop, called a DO-loop. 309

The form of a DO statement is: 311

DO s [,] i = e₁, e₂ [,e₃] 313

where: s   is the statement label of an executable 315
           statement. The statement identified by s, called 316
           the terminal statement of the DO-loop, must 317
           physically follow and appear in the same program 318
           unit as the DO statement. 319

       i   is the name of an integer, real, or double 321
           precision variable, called the DO-variable 322

       e₁, e₂, and e₃ are each an integer, real, or double 324
           precision expression. 325

The terminal statement of a DO-loop must not be an 327
unconditional GO TO, assigned GO TO, arithmetic IF, 328
block IF, ELSE IF, ELSE, END IF, RETURN, STOP, END, or DO 329
statement. If the terminal statement of a DO-loop is a 330
logical IF, it may contain any executable statement except a 331
DO, block IF, ELSE IF, ELSE, END IF, END, or another logical 332
IF statement. 333

11.10.1 Range of a DO-Loop 335

The range of a DO-loop consists of the executable statements 337
from and including the first executable statement following 338
the DO statement that specifies the DO-loop, to and 339
including the terminal statement of the DO-loop. 340

If a DO statement appears within the range of a DO-loop, the 342
range of the DO-loop specified by that DO statement must be 343
within the range of the outer DO-loop. More than one DO- 344
loop may have the same terminal statement. 345

If a DO statement appears within an IF-block, ELSE IF-block, 347
or ELSE-block, the range of that DO-loop must be contained 348
entirely within that IF-block, ELSE IF-block, or ELSE-block, 349
respectively. 350

If a block IF statement appears within the range of a DO- 352
loop, the corresponding END IF statement must also appear 353
within the range of that DO-loop. 354

FORTRAN/76 Full Language X3J3/76.3 (76-07-19)

CONTROL STATEMENTS Page 11-7

11.10.2 Active and Inactive DO-Loops 361

A DO-loop is either active or inactive. Initially inactive, 363
a DO-loop becomes active only when its DO statement is 364
executed. 365

Once active, the DO-loop becomes inactive only when: 367

(1) its iteration count is zero, 369

(2) its DO-variable becomes undefined or is redefined by 371
    means other than the incrementation described in 372
    11.10.7, 373

(3) a RETURN, STOP, or END statement is executed in its 375
    program unit, 376

(4) it is in the range of another DO-loop that becomes 378
    inactive, or 379

(5) it is in the range of another DO-loop whose DO 381
    statement is executed. 382

Note that transfer of control out of the range of a DO-loop 384
does not inactivate the DO-loop. However, the DO-loop 385
becomes inactive if the DO-variable becomes undefined or is 386
redefined outside the range. 387

When a DO-loop becomes inactive, the DO-variable of the DO- 389
loop retains its last defined value unless it has become 390
undefined. 391

11.10.3 Executing a DO Statement 393

The effect of executing a DO statement is to perform the 395
following steps in sequence: 396

(1) The initial parameter m₁, the terminal parameter m₂, 398
    and the incrementation parameter m₃ are established 399
    by evaluating e₁, e₂, and e₃, respectively, 400
    including, if necessary, conversion to the type of 401
    the DO-variable according to the rules for arithmetic 402
    conversion (Table 4). If e₃ does not appear, m₃ has 403
    a value of one. m₃ must not have a value of zero. 404

(2) The DO-variable becomes defined with the value of the 406
    initial parameter m₁. 407

(3) The iteration count is established and is the value 409
    of the expression 410

    MAX(INT((m₂ - m₁ + m₃)/m₃), 0) 412

    Note that the iteration count is zero whenever: 414

X3J3/76.3 (76-07-19) FORTRAN/76 Full Language

$m_1 > m_2$ and $m_3 > 0$, or   421

$m_1 < m_2$ and $m_3 < 0$.   423

At the completion of execution of the DO statement, loop   425
control processing begins.   426

11.10.4 Loop Control Processing   428 |

Loop control processing determines if further execution of   430
the range of the DO-loop is required. The iteration count is 431
tested. If it is not zero, execution of the first statement   432
in the range of the DO-loop begins. If the iteration count   433
is zero, the DO-loop becomes inactive. If, as a result, all   434
of the DO-loops sharing the terminal statement of this DO-   435
loop are inactive, normal execution continues with execution 436
of the next executable statement following the terminal   437
statement. However, if some of the DO-loops sharing the   438
terminal statement are active, execution continues with   439
incrementation processing, as described below.   440

11.10.5 Execution of the Range   442 |

Statements in the range of a DO-loop are executed until the   444
terminal statement is reached. Except by the incrementation   445
described in 11.10.7, the DO-variable of the DO-loop may   446 |
neither be redefined nor become undefined during execution   447
of the range of the DO-loop.   448

11.10.6 Terminal Statement Execution   450 |

Execution of the terminal statement occurs as a result of   452
the normal execution sequence or as a result of transfer of   453
control, subject to the restrictions in 11.10.8. Unless   454 |
execution of the terminal statement results in a transfer of 455
control, execution then continues with incrementation   456
processing, as described below.   457

11.10.7 Incrementation Processing   459 |

Incrementation processing has the effect of the following   461
steps performed in sequence: 462

(1) The DO-variable, and the incrementation parameter of   464
    the active DO-loop whose DO statement was most   465
    recently executed, are selected for processing.   466

(2) The value of the DO-variable is incremented by the   468
    value of the incrementation parameter m_3.   469

(3) The iteration count is decremented by one.   471

(4) Execution continues with loop control processing   473
    (11.10.4) of the same DO-loop whose iteration count   474 |
    was decremented.   475

An example illustrates the above: 481

    N=0   483
    DO 100 I=1,10   484
    J=I   485
    DO 100 K=1,5   486
    L=K   487
100 N=N+1   488
101 CONTINUE   489

After execution of the above statements and at the execution 491
of the CONTINUE statement, I=11, J=10, K=6, L=5, and N=50.   492

Also consider the following example: 494

    N=0   496
    DO 200 I=1,10   497
    J=I   498
    DO 200 K=5,1   499
    L=K   500
200 N=N+1   501
201 CONTINUE   502

After execution of the above statements and at the execution 504
of the CONTINUE statement, I=11, J=10, K=5, and N=0. L is   505
not defined by the above statements.   506

11.10.8 Transfer into the Range of a DO-Loop   508 |

Transfer of control into the range of an inactive DO-loop is 510
not permitted. Transfer of control to any executable   511
statement in the range of an active DO-loop is permitted   512
unless the statement is also in the range of an inactive DO- 513
loop.   514

11.11 CONTINUE Statement   517 |

The form of a CONTINUE statement is: 519

        CONTINUE   521

Execution of a CONTINUE statement has no effect.   523

If the CONTINUE statement is not the terminal statement of a 525
DO-loop, normal execution sequence continues. If the   526
CONTINUE statement is the terminal statement of a DO-loop,   527
the next statement executed depends on the result of the DO- 528
loop incrementation processing (11.10.7).   529 |

11.12 STOP Statement   532 |

The form of a STOP statement is: 534

        STOP [n]   536

Page 11-10 CONTROL STATEMENTS

where n is a string of not more than five digits, or is a   541
character constant.   542

Execution of a STOP statement causes termination of   544
execution of the executable program. At the time of   545
termination, the digit string or character constant is   546
accessible.   547

11.13 PAUSE Statement   550 |

The form of a PAUSE statement is: 552

              PAUSE [n]   554

where n is a string of not more than five digits, or is a   556
character constant.   557

Execution of a PAUSE statement causes a cessation of   559
execution of the executable program. Execution must be   560
resumable. At the time of cessation of execution, the digit   561
string or character constant is accessible. Resumption of   562
execution is not under control of the program. If execution   563
is resumed, the normal execution sequence is continued.   564

11.14 END Statement   567 |

The END statement indicates the end of the sequence of   569
statements and comment lines of a program unit (3.5). If   570
executed in a   subprogram, it has the effect of a RETURN   571
statement (15.8).   If executed in a main program, it   572
terminates the execution of the executable program.   573

The form of an END statement is: 575

              END   577

An END statement is written only in columns 7 through 72 of   579
an initial line. An END statement must not be continued.   580
No other statement in a program unit may have an initial   581
line that appears to be an END statement.   582

The last line of every program unit must be an END   584
statement.   585

FORTRAN/76 Full Language X3J3/76.3 (76-07-19)